Technical Report 70-121 July 1970
NGL-21-002-008

ELIMINATION PROCEDURES FOR SPARSE SYMMETRIC LINEAR SYSTEMS OF A SPECIAL STRUCTURE

by

Jitka Segethova

UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER
COLLEGE PARK, MARYLAND

FACILITY FORM 602

N71-70616 (ACCESSION NUMBER)	(THRU)
24 (PAGES)	(CODE)
CR-116168 (NASA CR OR TMX OR AD NUMBER)	(CATEGORY)

FEB 1971
RECEIVED
NASA STI FACILITY
INPUT BRANCH

Technical Report 70-121 July 1970
NGL-21-002-008

ELIMINATION PROCEDURES FOR SPARSE SYMMETRIC LINEAR SYSTEMS OF A SPECIAL STRUCTURE

by

Jitka Segethova

This research was sponsored in part by the National Aeronautics and Space Administration, Grant NGL 21-002-008, to the Computer Science Center of the University of Maryland.

INTRODUCTION

The systems of linear algebraic equations which arise in solving differential equations by finite element methods usually have matrices which are sparse and of a certain regular structure. For the solution of such a system by elimination, it is desirable to use these properties of the corresponding matrix. More specifically, we wish to find an ordering of the rows and columns and an algorithm for solving the system such that the storage requirements and the number of operations performed during the elimination are minimized. This paper discusses the problem of finding such a permutation of rows and columns and an algorithm for this type of ordered system of equations.

There exist some approaches to this problem in which the sparsity is used to a certain extent. One of them is very general in that the optimal (or nearly optimal) ordering is sought and that the algorithm for solving the ordered system treats the matrix element by element to ensure that only necessary operations are performed. This case and the case of bandmatrices are compared with respect to the regularity of the structure of a given matrix before and after ordering. In connection with this comparison another approach is introduced. A type of matrix more general than a bandmatrix is considered as well as the means to order the rows and columns to get this form. Then examples of matrices reordered by the given procedure together with the corresponding results are discussed.

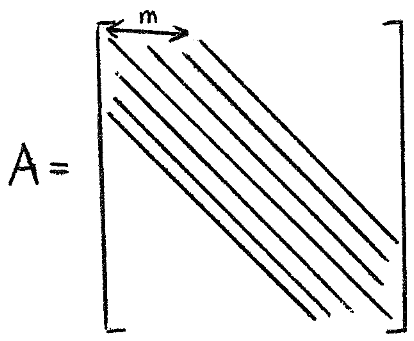
1. Problems of elimination for sparse systems.

In solving differential equations by the finite element or finite difference method we usually obtain systems of linear algebraic equations, with large sparse matrices of certain regular zero-nonzero structures, which must be solved. When solving these systems by Gauss elimination we will want to use the sparsity as well as the regularity of zero-nonzero structure of a given matrix. Our aim is to reduce storage requirements and the number of operations performed during the elimination. Fewer operations take less time and result in less roundoff error.

The first problem is, given an ordering of rows and columns, we want to operate only on such elements which are involved and changed in the elimination process itself. The second problem is to find an ordering of rows and columns of a given matrix so that the number of nonzero elements created in the forward course of Gauss elimination as well as the number of operations performed is minimal for a given matrix. Further we should consider the structure of a given matrix before as well as after reordering it. Usually we solve many systems with matrices of the same structure which differ from each other only by some parameter, for example a mesh size h . It is desirable to find a general rule for ordering such classes of matrices.

2. A brief survey of techniques used.

One of the simplest ways to solve these problems is to consider certain matrices as bandmatrices. Let us have an nxn symmetric bandmatrix.

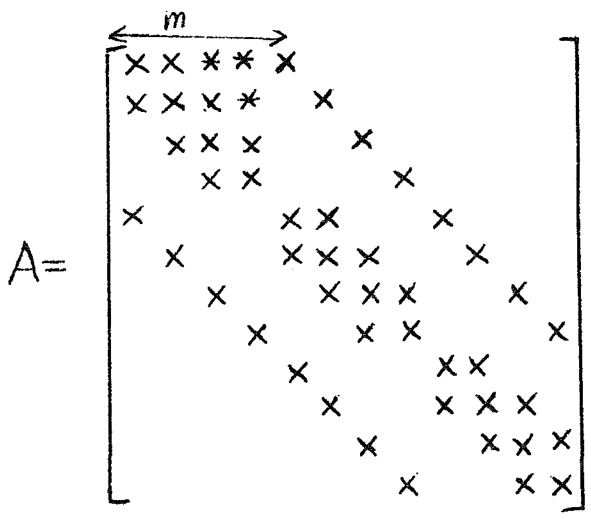


Let us suppose that the nonzero elements appear only in the indicated band, i.e. $a_{ij} \neq 0 \Rightarrow |i-j| \leq m$. Then it is sufficient to store and operate only on the elements which are within the band or in the symmetric case in the upper half of the band.

The number of multiplications performed during the forward course of Gauss elimination depends upon m $\left(\text{it is} \sim \frac{m(m+3)}{2} n\right)$ as well as the storage requirements.

The bandmatrices offer important advantages. First, for a symmetric bandmatrix A all the "fill in" due to elimination is within the original band. On the other hand all the elements which are within the band are operated on, even those which are possibly not involved in the elimination process itself. (For example the elements denoted by * in the example 2.1.)

Example 2.1.



Secondly, the elimination scheme for a symmetric bandmatrix is particularly simple. Also data handling is very simple because the zero-nonzero structure of a matrix is described in terms of m , n .

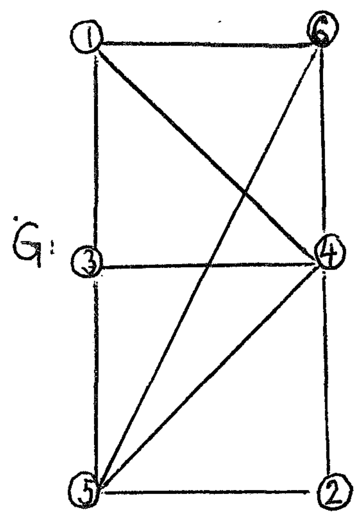
A change of the parameter h will result only in a change of m , n . But very often it is useful to find a permutation of rows and columns to reduce the width of a bandmatrix. This is a complicated problem and it has been discussed by E. Cuthill, J. McKee [1], G. G. Alway, D. W. Martin [2], R. Rosen [3], and others.

The most efficient way for solving the sparse linear algebraic systems (from the point of view of minimization of number of operations and storage) is to find an ordering of rows and columns which minimizes the number of operations and the number of nonzero elements created. Problems of finding a so-called optimal or nearly-optimal ordering have been discussed by many authors; S. Parter [4], R. P. Tewarson [5], W. R. Spillers, N. Hickerson [7], and others.

Let us have a symmetric matrix A with the graph G and let us suppose that the elimination can be performed in the arbitrary ordering of rows and columns. (This assumption is satisfied, for example, by symmetric positive definite matrices.) Also let A be irreducible (then G is connected).

Example 2.2.

$$A = \begin{bmatrix} \times & 0 & \times & \times & 0 & \times \\ 0 & \times & 0 & \times & \times & 0 \\ \times & 0 & \times & \times & \times & 0 \\ \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & 0 \\ \times & 0 & 0 & \times & 0 & \times \end{bmatrix}$$



Let us solve the system of linear algebraic equations:

$$Ax = y \quad .$$

Eliminating the unknown x_i (or the node i in the graph) we must perform $\frac{b_i(b_i + 3)}{3}$ multiplications, where b_i is a number of nodes which are connected to the node i (or the number of nonzero elements in the row $\underline{i}$).

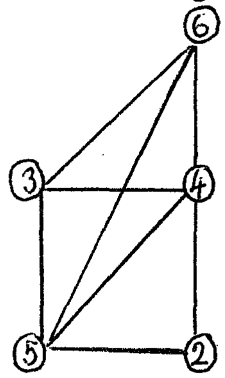
Let N(i) be a set of nodes $m \in G$ which are connected with the node i by an edge in the graph G (or $N(i) = \{m: a_{im} \neq 0\}$).

Eliminating the node i from the graph G we get the graph G_i, where $i \notin G_i$ and all the nodes $m \in N(i)$ are pairwise connected. Therefore eliminating the node i from G (the unknown x_i) we obtain z_i new edges in the graph (z_i new nonzero elements are created since, by symmetry, we consider only the upper triangular matrix).

In the example 2.2:

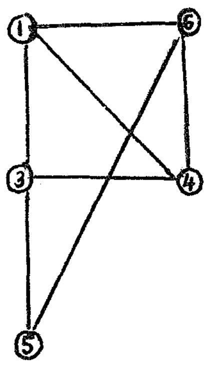
Eliminating the node 1 we get the graph G_1.

G_1:



where the edge connecting 3 , 6 has been added. Eliminating the node 2 we get the graph G_1'.

G_1':



where no edge has been added into the original graph. In practice the nearly optimal ordering is usually found by local minimization of degree (b_i) or fill in (z_i) in every step. These algorithms are very simple and often very efficient.

Having a nearly optimal or optimal ordering we must then solve the ordered system by an algorithm operating only on those elements which are actually changed by the elimination process (e.g., see F. Gustavson [6]). Therefore, the matrix must be treated element by element and the zero-nonzero structure of such a matrix is described by the positions of every nonzero element. Therefore, the use of this program can entail a considerable amount of work. This approach is very general (further this algorithm for solving the system by elimination is called "general algorithm") and it is particularly suitable for those matrices whose nonzero elements occur in no regular structure.

3. The pipematrices.

Let us try another approach. Let us consider a symmetric matrix of the form:

$$A = \begin{bmatrix} x & x & 0 & x & x & 0 & 0 \\ x & x & 0 & x & x & 0 & 0 \\ 0 & 0 & x & x & x & x & 0 \\ x & x & x & x & x & x & 0 \\ x & x & x & x & x & x & x \\ 0 & 0 & x & x & x & x & x \\ 0 & 0 & 0 & 0 & x & x & x \end{bmatrix}$$

The nonzero elements are indicated by the x's. We shall call this matrix a "pipematrix." In the elimination process applied to a system with a matrix of this form only the elements of the marked parts of columns (the so-called "pipes") must be operated on. No other element is changed by the elimination process. If the pipes involve only nonzero elements, no nonzero element is created during the elimination, i.e. "fill in" equals zero. If some zeroes appear in the pipes, nonzero elements may be created.

The algorithm for solving the linear algebraic systems with pipematrices treats the elements in the pipes.

Let us have a matrix A in the example 3.1.

Example 3.1.

$$A = \begin{bmatrix} x & 0 & x & x & 0 \\ 0 & x & x & 0 & 0 \\ x & x & x & 0 & 0 \\ x & 0 & 0 & x & x \\ 0 & 0 & 0 & x & x \end{bmatrix}$$

Solving the system by the general algorithm, where the elements of a matrix are treated individually, a nonzero element is created in position 3, 4, however the element a_{24} remains zero (is not operated on) so that fill is one element (a_{34}). Considering the matrix as a pipematrix, we operate with the whole pipes, i.e., with elements a_{11};

a_{22}; a_{13}, a_{23}, a_{33}; a_{14}, $\underline{a_{24}}$, a_{34}, a_{44}; a_{45}, a_{55}. Therefore, in general, when the pipes involve some zeroes we may perform operations which are not necessary (as occurred in the example 2.1 of a bandmatrix).

It is more general to consider an arbitrary matrix as a pipematrix rather than a bandmatrix. If we consider an arbitrary matrix as a pipematrix we generally perform fewer or at most the same number of operations as if we consider it as a bandmatrix.

On the other hand it is less general to treat a matrix as a pipematrix than to treat it by the general algorithm — element by element. The zero-nonzero structure of a pipematrix is described by the positions of whole pipes which makes this algorithm as well as its use (handling data) simpler than the general one.

We can describe the zero-nonzero structure of a given pipematrix by n parameters m_ℓ , where (m_ℓ , ℓ) is the first nonzero position in the pipe ending at (ℓ, ℓ). In the example 3.1:

$$m_1 = 1 , \quad m_2 = 2 , \quad m_3 = 1 , \quad m_4 = 1 , \quad m_5 = 4 .$$

Let us have a pipematrix A . Let q_i be the number of nonzero elements (or elements which are considered nonzero because they appear in the pipes) in the row i (in the upper triangle). In the example 3.1: $q_1 = 2$, $q_2 = 2$, $q_3 = 2$, $q_4 = 1$, $q_5 = 0$. Then the number of multiplications performed during the forward course of Gauss elimination is

$$\sum_{i=1}^{n} \frac{q_i(q_i + 3)}{2} \quad .$$

It is evident that the number of operations performed during the elimination process for a pipematrix depends on the number of pipes which intersect a particular row and on the length of every pipe.

Now the problem arises to find a permutation by which the matrix would be reordered into the form of a pipematrix whose pipes include as few zero elements as possible.

Let A be a symmetric irreducible matrix which can be eliminated with an arbitrary order of rows and columns. Suppose A can be permuted into the pipematrix form with full pipes. Let G_o be the graph of the matrix A. The following procedure gives the permutation by which the matrix is reordered into the form with full pipes.

<u>Procedure 1</u>

<u>Step 1</u>

1. Set $S_1^1 = \{i : z_i = 0\}$

   (According to the assumptions about the matrix A there exists at least one node i for which $z_i = 0$.)

2. Either

   A. Select $i \in S_1^1$ arbitrarily or

   B. Set $S_2^1 = \left\{ i : i \in S_1^1 \ , \ b_i = \min_{k \in S_1^1} b_k \right\}$

   and select $i \in S_2^1$ arbitrarily.

3. Set $S^1 = \{m: m \in N(i)\}$ $(= \{m: a_{im} \neq 0\})$

   Eliminate the node i from the graph G_o .

<u>Step $k = 2, \ldots, n-1$</u>

1. Set $R^k = \left\{j: j \in N(m) \text{ for all } m \in S^{k-1}\right\}$

   $(R^k \neq \emptyset$ because $S^{k-1} \subset R^k)$

2. Set $S_1^k = \{j: j \in R^k , z_j = 0\}$

   If $S_1^k = \emptyset$ delete i from S_i^{k-1} and repeat the step $k-1$.

3. If $S_1^k \neq \emptyset$ then corresponding to the choice of either 2A or 2B in step 1, choose either

   A. Select $i \in S_1^k$ arbitrarily or

   B. Set $S_2^k = \left\{ j: j \in S_1^k , j \in S^{k-p_j} , p_j = \max_{q \in S_1^k} p_q \right\}$

   $(= \{j: j$ is an index of the longest pipe$\}$

   Set $S_3^k = \left\{ j: j \in S_2^k , b_j = \min_{q \in S_2^k} b_q \right\}$

   Select $i \in S_3^k$ arbitrarily

4. Set $S^k = \{m: m \in N(i)\}$

   Eliminate the node i from the graph G_{k-1} .

<u>Remark</u>

If we choose the alternative 2B instead of 2A in step 1 and 3B instead of 3A we may require less time to complete the process.

With this algorithm the matrix is permuted into the pipematrix form with full pipes under the assumptions mentioned above.

Procedure 2

Step 1

1. Set $S_1^1 = \left\{ i:\ z_i = \min\limits_{k=\{1,\dots,n\}} z_k \right\}$

2. Either

   A. Select $i \in S_1^1$ arbitrarily or

   B. Set $S_2^1 = \left\{ i:\ i \in S_1^1\ ,\ b_i = \min\limits_{k \in S_1^1} b_k \right\}$

   Select $i \in S_2^1$ arbitrarily.

3. Set $S^1 = \{m:\ m \in N(i)\}$

   Eliminate the node i from the graph G_o (or, equivalently, connect the nodes $m \in N(i)$ pairwise).

Step $k = 2, \dots, n-1$

1. Set $R^k = \left\{ j:\ j \in N(m)\ \text{ for all }\ m \in S^{k-1} \right\}$
2. Set $S_1^k = \left\{ j:\ j \in R^k\ ,\ z_j = \min\limits_{q \in R^k} z_q \right\}$

3. Corresponding to the choice of 2A or 2B above, choose either

   A. Select $i \in S_1^k$ arbitrarily or

   B. Set $S_2^k = \left\{ j:\ j \in S_1^k\ ,\ j \in S^{k-pj}\ ,\ \text{where }\ pj = \max\limits_{q \in S_1^k} p_q \right\}$

   Set $S_3^k = \left\{ j:\ j \in S_2^k\ ;\ bj = \min\limits_{q \in S_2^k} bq \right\}$

   Select $i \in S_3^k$ arbitrarily.

4. Set $S^k = \{m:\ m \in N(i)\}$.

   Eliminate the node $\underline{i}$ from the graph G_{k-1} .

Remark

If we choose the alternatives 2B instead of 2A and 3B instead of 3A we sometimes obtain a better ordering.

The procedures shown above have these important properties:

1. If the matrix can be permuted into the form with full pipes then procedure 1 gives the corresponding permutation.
2. If procedure 2 is applied to the matrices which cannot be permuted into the form with full pipes it may give an acceptable ordering in some examples as will be shown in Chapter 4.
3. Procedure 2 is not very time consuming because only a certain set of nodes is tested in each step.
4. The only time consuming part of procedure 1 may be part 2 in the steps $k = 2, \ldots, n - 1$ where it can theoretically happen that we go back to the very beginning of the procedure several times. However, in the computations performed where a matrix was permuted into the form with full pipes only procedure 2 was used and, therefore, this problem did not arise.
5. Assume we have an arbitrary matrix ordered by the permutation given by either procedure 1 or 2. Let us solve the system of linear algebraic equations by the algorithm for pipematrices. Then the same number of operations are required as in the general algorithm where the matrix is treated element by element (e.g., F. Gustavson [6]). This follows from the procedure itself.

4. Examples and results.

Procedure 2 was applied to several types of matrices. The results were very interesting. Let us show two typical examples where the procedure yields a satisfactory ordering. In procedure 2, the parts 2B and 3B were used (instead of 2A and 3A, respectively). In the parts 2B and 3B where "Select $i \in S_2^1$ (S_3^k) arbitrarily" is recommended, the node with the lowest number in the original ordering was selected.

Example 4.1

(The example of a matrix that can be reordered into the pipematrix form with full pipes.)

The matrix which arises from mesh refinement in one dimension when the solution of a certain boundary value problem is approximated by one type of hill functions (See: I. Babuska [8], [9]) has a graph of the type in the Fig. 4.1.

The corresponding matrix is given in Fig. 4.2. Procedure 2 gives the ordering in Fig. 4.3 and the reordered matrix is A_o in Fig. 4.4. Eliminating the system with the matrix A_o (by the elimination procedure for pipematrices) we get zero fill.

We can obtain several orderings by which the matrix is reordered into the pipematrix form with full pipes by starting with various original orderings.

Example 4.2

Suppose we have a matrix with graph G in the figure 4.5 (which arises from using the five point difference formula in approximating certain boundary value problems on an L-shaped domain). If we number the nodes in the so-called natural ordering (See Fig. 4.5) we obtain a bandmatrix with $m = m_4$.

Let us number the nodes by procedure 2. We then have the ordering indicated in Fig. 4.6. With this ordering, the number of operations performed during the elimination in part I (see Fig. 4.5) is less than or equal to that for the bandmatrix with $m = \min (m_1, m_2 + m_3)$ and in part II with $m = \min (m_3, m_4)$. This can result in a substantial time and operations reduction. The same results are valid for various types of L-shaped domains which differ only in the ratios of m_1, m_2, m_3, m_4 . Moreover, having matrices with the graph in Fig. 4.5 where only h (a mesh size) is different, it is not necessary to seek an ordering for every matrix because the procedure gives a general rule for orderings with various magnitudes of h . For example, let us consider the L-shaped domain above. Then for every part of the domain there is a formula describing the relation between the coordinates of the node (x, y) in the net (it is also the node in the graph of the matrix) and the number $N(x, y)$ in the ordering. For example in part 1 (See Fig. 4.7)

$$N(x,y) = \frac{1}{2}\left(\frac{x^2}{h^2} + \frac{y^2}{h^2} + \frac{x}{h} + \frac{y}{h}\right) + \frac{xy}{h^2} + \frac{x}{h} + \frac{y - x}{h} \delta_{ox} + \delta_{oy}$$

(where $\delta_{ox} = 1$ if $x = 0$ and $\delta_{oy} = 0$ if $y = 0$)

is valid. Similar formulas are valid for the other parts of the domain.

Let us have a matrix with graph G in Fig. 4.5. Let us permute the rows and columns using the ordering given by procedure 2 and let us solve the system by the algorithm for pipematrices. Also let us permute the rows and columns in the ordering given by the "minimal degree algorithm" and solve the system by the general algorithm. Then we have the following results:

$$(m_1 = 5,\ m_2 = 5,\ m_3 = 5,\ m_4 = 12,\ n = 80)$$

1. Ordered in the so-called "natural ordering" we get the bandmatrix with $m = 12$. Solved by the algorithm for bandmatrices $80 \times 13 = 1040$ elements are treated.
2. Ordered by procedure 2 and solved by the algorithm for pipematrices, fill is 230.
3. Ordered in the ordering given by the "minimal degree algorithm" and solved by the general algorithm, fill is 188.

In comparison with 1 the fills in 2 and 3 are approximately the same, but in case 3 we require the general algorithm for solving the system.

The technique where the matrix is permuted by the procedure 1 or 2 into the pipematrix form and the system is solved in the corresponding way is advantageous if applied to certain matrices (as the example 4.1, 4.2) in comparison with both of the other approaches mentioned in Chapter 2.

The algorithm for solving the system with a pipematrix and, in particular, its operation (handling input data) are simpler than the general algorithm and its application. On the other hand a pipematrix form is more general than a bandmatrix form, which, in turn, may be an advantage; particularly if the ordering has been found for a set of matrices with the same zero-nonzero structure as in the example 4.2.

G :

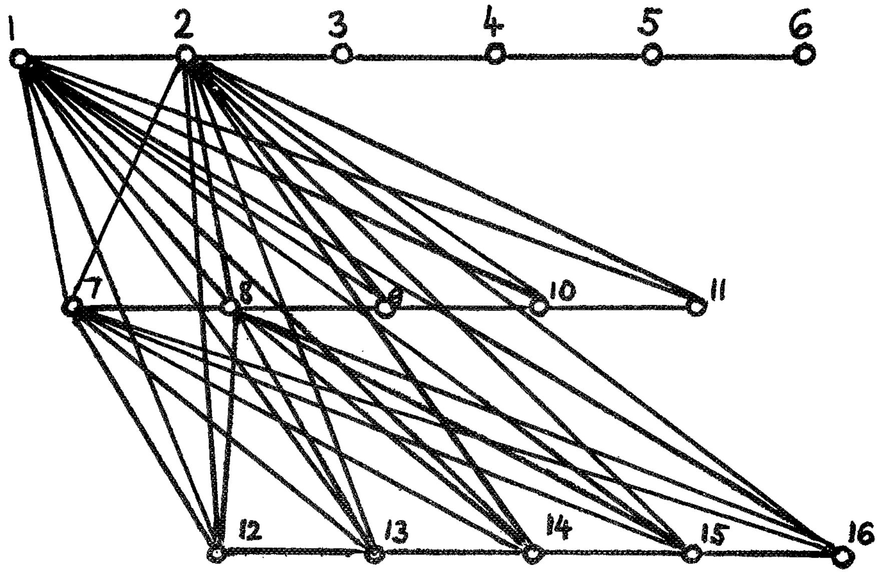


Fig. 4.1.

$$A=\begin{bmatrix}
\times&\times&&&&&\times&\times&\times&\times&\times&\times&\times&\times&\times&\times\\
\times&\times&\times&&&&\times&\times&\times&\times&\times&\times&\times&\times&\times&\times\\
&\times&\times&\times&&&&&&&&&&&&\\
&&\times&\times&\times&&&&&&&&&&&\\
&&&\times&\times&\times&&&&&&&&&&\\
&&&&\times&\times&&&&&&&&&&\\
\times&\times&&&&&\times&\times&&&&\times&\times&\times&\times&\times\\
\times&\times&&&&&\times&\times&\times&&&\times&\times&\times&\times&\times\\
\times&\times&&&&&&\times&\times&\times&&&&&&\\
\times&\times&&&&&&&\times&\times&\times&&&&&\\
\times&\times&&&&&&&&\times&\times&&&&&\\
\times&\times&&&&&\times&\times&&&&\times&\times&&&\\
\times&\times&&&&&\times&\times&&&&\times&\times&\times&&\\
\times&\times&&&&&\times&\times&&&&&\times&\times&\times&\\
\times&\times&&&&&\times&\times&&&&&&\times&\times&\times\\
\times&\times&&&&&\times&\times&&&&&&&\times&\times
\end{bmatrix}.$$

Fig. 4.2.

G_{10}:

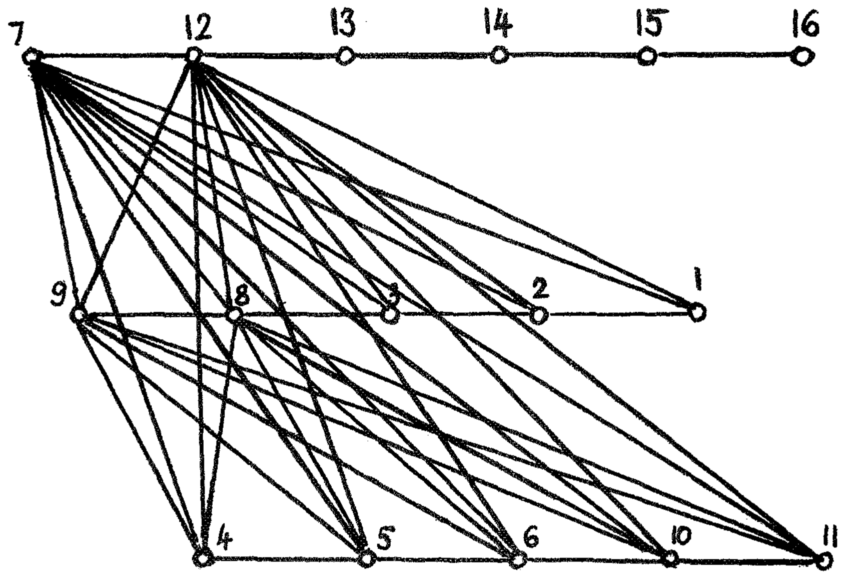


Fig. 4.3.

$$A_0 = \begin{bmatrix}
\times & \times & & & & & \times & & & & & \times & & & & \\
\times & \times & \times & & & & \times & & & & & \times & & & & \\
& \times & \times & & & & \times & \times & & & & \times & & & & \\
& & & \times & \times & & \times & \times & \times & & & \times & & & & \\
& & & \times & \times & \times & \times & \times & \times & & & \times & & & & \\
& & & & \times & \times & \times & \times & \times & \times & & \times & & & & \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & & & & \\
& & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & & & & \\
& & & \times & \times & \times & \times & \times & \times & \times & \times & \times & & & & \\
& & & & & \times & \times & \times & \times & \times & \times & \times & & & & \\
& & & & & & \times & \times & \times & \times & \times & \times & & & & \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & & & \\
& & & & & & & & & & & \times & \times & \times & & \\
& & & & & & & & & & & & \times & \times & \times & \\
& & & & & & & & & & & & & \times & \times & \times \\
& & & & & & & & & & & & & & \times & \times
\end{bmatrix}$$

Fig. 4.4.

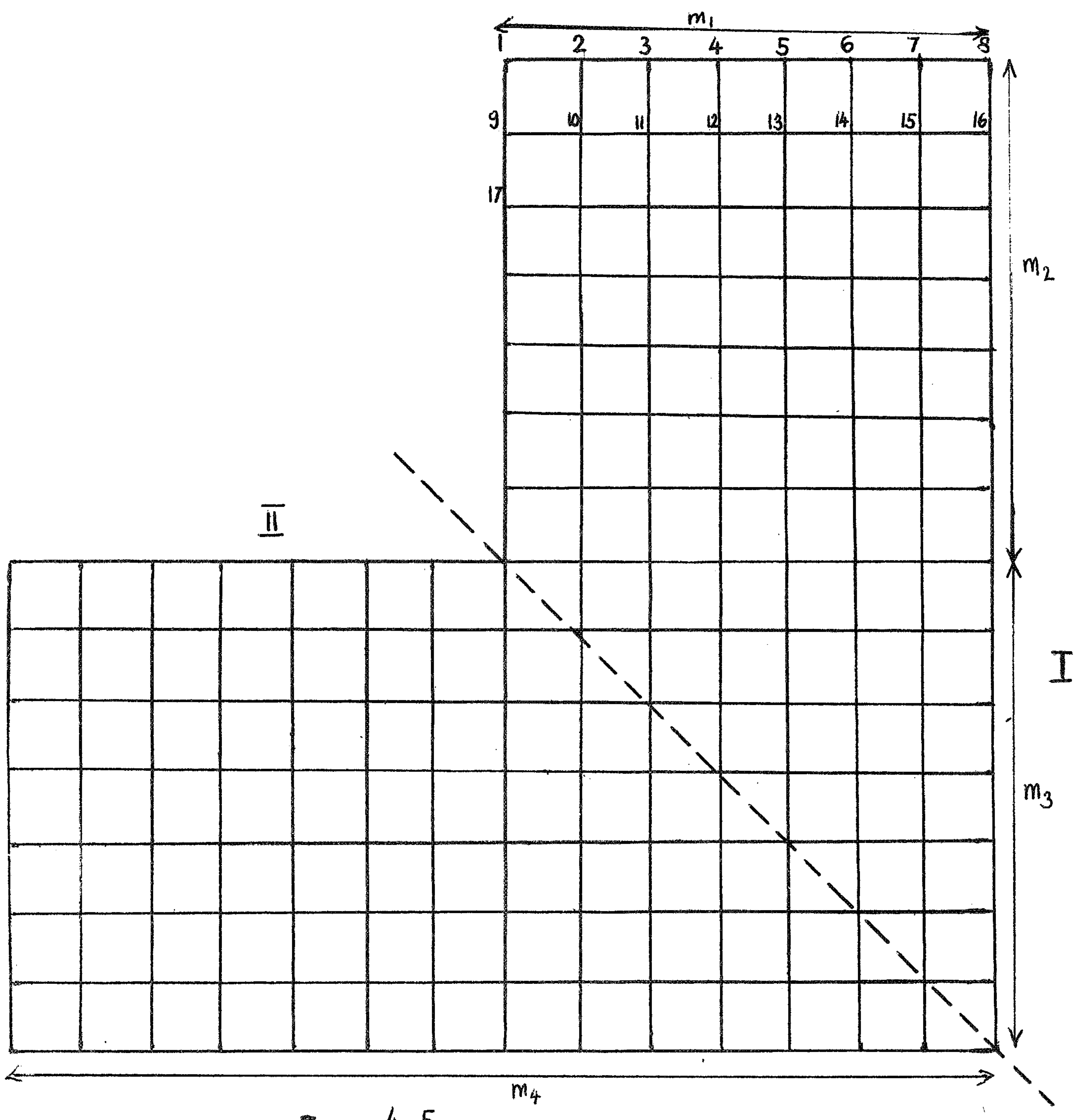


Fig. 4.5.

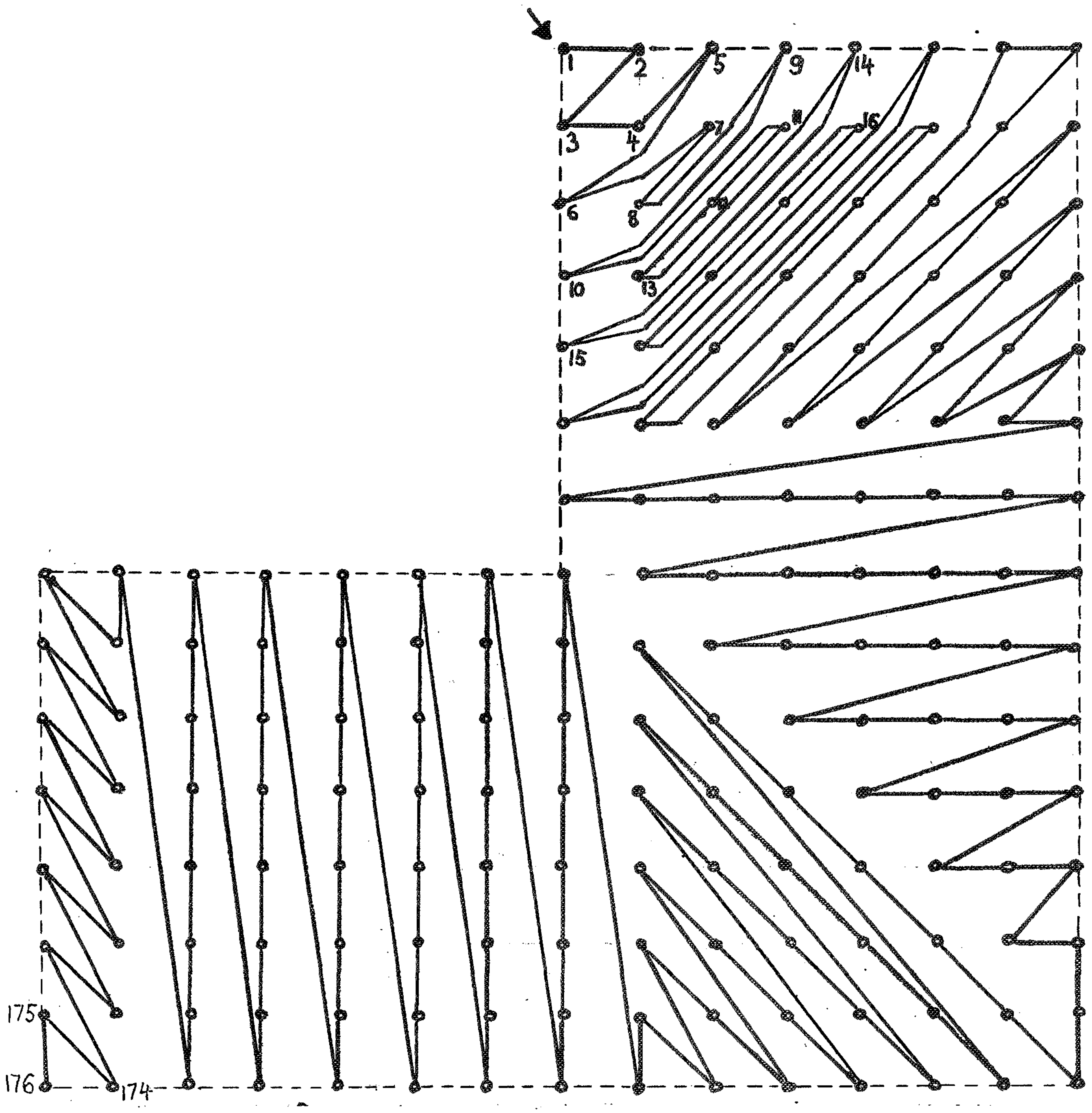


Fig. 4.6.

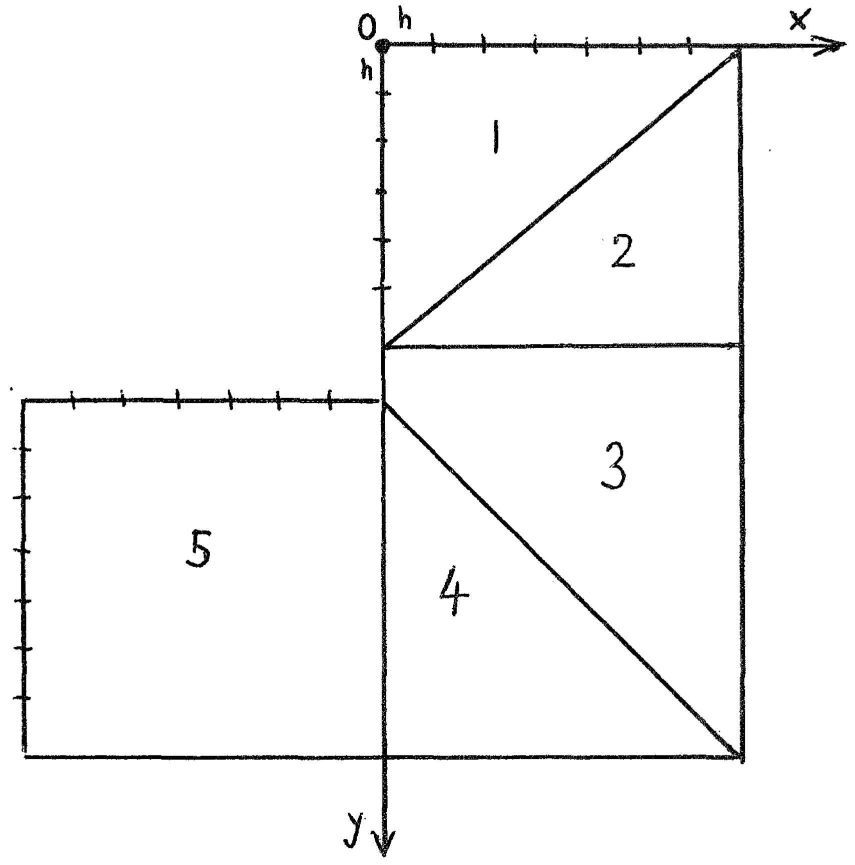


Fig. 4.7.

References

[1] E. Cuthill, J. McKee, Reducing the bandwidth of sparse symmetric matrices, 1969 Summer Natl. Acm Meeting Proceedings.

[2] G. G. Alway, D. W. Martin, An algorithm for reducing the bandwidth of a matrix of symmetric configuration, Computer J. 8 (1965/66), 264-72.

[3] R. Rosen, Matrix bandwidth minimization, ACM National Conference, Las Vegas, Nevada, 1968.

[4] S. Parter, The use of linear graphs in Gauss elimination, SIAM Rev. 3 (1961), 119-30.

[5] R. P. Tewarson, The Gaussian Elimination and Sparse Systems, Proc. of the Symposium on Sparse Matrices and Their Appl., IBM Watson Research Center, 1968.

[6] F. G. Gustavson, W. M. Liniger, R. A. Willoughby, Symbolic Generation of an Optimal Crout Algorithm for Sparse Systems of Linear Equations, Proc. of the Symposium on Sparse Matr. and Their Appl., IBM Watson Res. Center, 1968.

[7] W. R. Spillers, N. Hickerson, Optimal elimination for sparse symmetric systems as a graph problem, Quar. Appl. Math. 26 (1968) 425-32.

[8] I. Babuska, Finite element method for domains with corners, to appear.

[9] I. Babuska, The rate of convergence for the finite element method, to appear.